A web-based intelligent educational system for PROLOG

Christoph Peylo – Tobias Thelen – Claus Rollinger - Helmar Gust
{Christoph.Peylo|Tobias.Thelen|Claus.Rollinger|Helmar.Gust}uos.de

Institute for Semantic Information Processing
University of Osnabrück

April 2000

1 Abstract

In this paper we introduce a web-based intelligent educational system for PROLOG. In the first section we show its relationship to current research on web-based educational systems. In the second part we give a survey on the design of the system and the kind of support it offers to both students and teachers. The third part explains the technologies that are deployed for intelligent user support and user modelling. In the fourth part we discuss our experiences with the system during the last terms. Finally, we give an outlook on future work.

2 Introduction

Currently, web-based educational systems are a challenging research and developing area. Benefits of web based education are independence of teaching and learning with respect to time and space. Course ware installed and maintained in one place may be used by a huge number of users all over the world. The World Wide Web offers an unprecedented opportunity for addressing large audiences with intelligent tutoring technology and thus giving rise to various endeavours in this field.

In this paper we will introduce the "Virtual Campus PROLOG Tutor", a web-based educational system that has been accomplished in the course of the "Virtual Campus" (VC) project. The VC is a joint effort of the universities of Hannover, Osnabrück and Hildesheim to integrate Internet technologies into education and to develop lectures and environments for several courses accessible through the Internet. This educational setting containing both local and remote resources may be regarded as a virtual campus, i.e. a virtual university where people meet and communicate mediated by net-based technology.

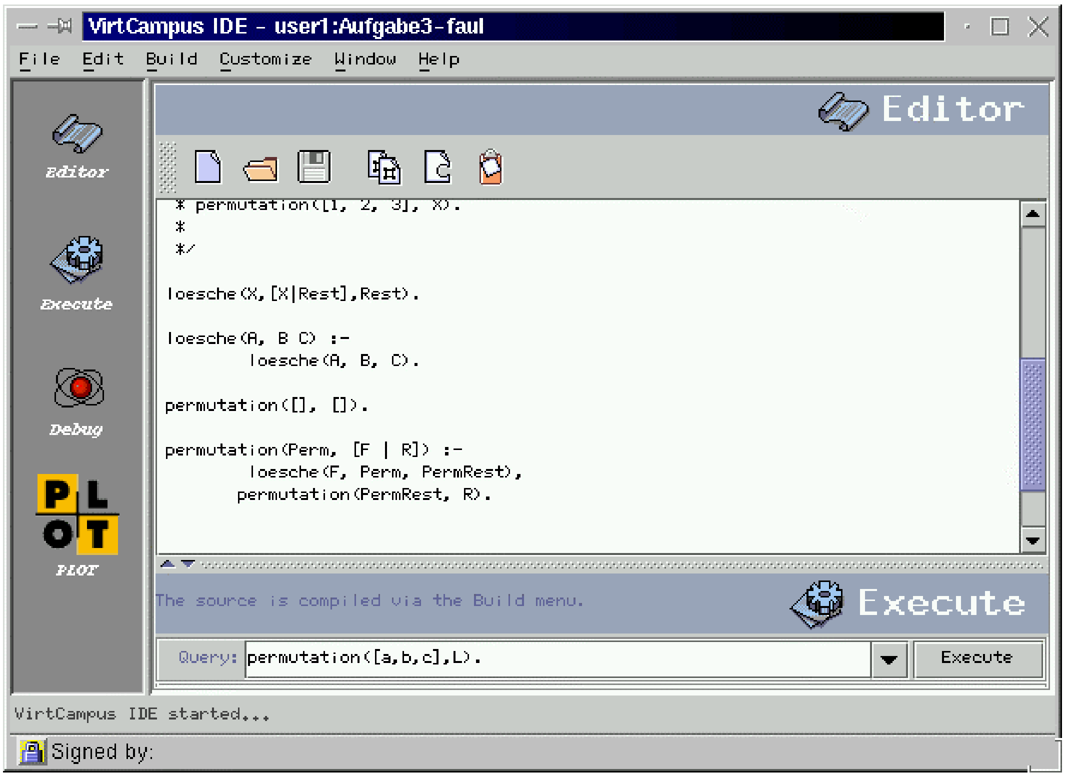

Figure: Screenshot: editor of the VC PROLOG tutor

After a short survey on the educational setting and the VC PROLOG tutor's features we will go into more detail concerning systems' architecture and the deployed technologies.

3 The VC PROLOG tutor

Traditionally, academic teaching and learning is situated in a distinct social and spatial context. Studying does not only involve attending seminars and lectures but making use of libraries and interaction with fellow students and teachers. The traditional ITS paradigm is rooted in this setting, trying to bring aspects of the one-to-one tutoring experience to a larger audience [Blo84]. The ITS approach puts the focus on tutoring, other aspects that play a role in university studies like retrieving of literature and learning materials are not drawn into account. Thus, almost no ITS systems includes educational material itself [Bru99]. The situation is different in the context of web-based education and more similar to distance education. Students who want to benefit from universities' remote offerings need a different kind of support than students who are physically present at campus. All resources for Internet lessons have to be available in the net and adopted to the needs of that kind of student. Knowledge which used to be taught in a more or less implicit way in traditional courses has to be made explicit. Furthermore, topics may be introduced and taught in traditional lecturing in a certain order depending on the background knowledge of the students which is roughly the same in a certain class. Since learning in the Internet is a more individualized learning, the whole knowledge, which is relevant to understand a distinct domain, has to be organized in a way that provides the student with a comprehensive model of that domain. Thus, technologies for representing and presenting domain knowledge, are becoming very important to guide the student through the hyper-space of available education material.

3.1 Educational Background

The VC PROLOG tutor is used at the university of Osnabrück for seminars and tutorials accompanying PROLOG lectures. During seminars and tutorials human tutors are present to offer additional support and advice. At the same time the system is intended to meet the requirements of remote students who want to learn PROLOG via the Internet in a self-directed way.

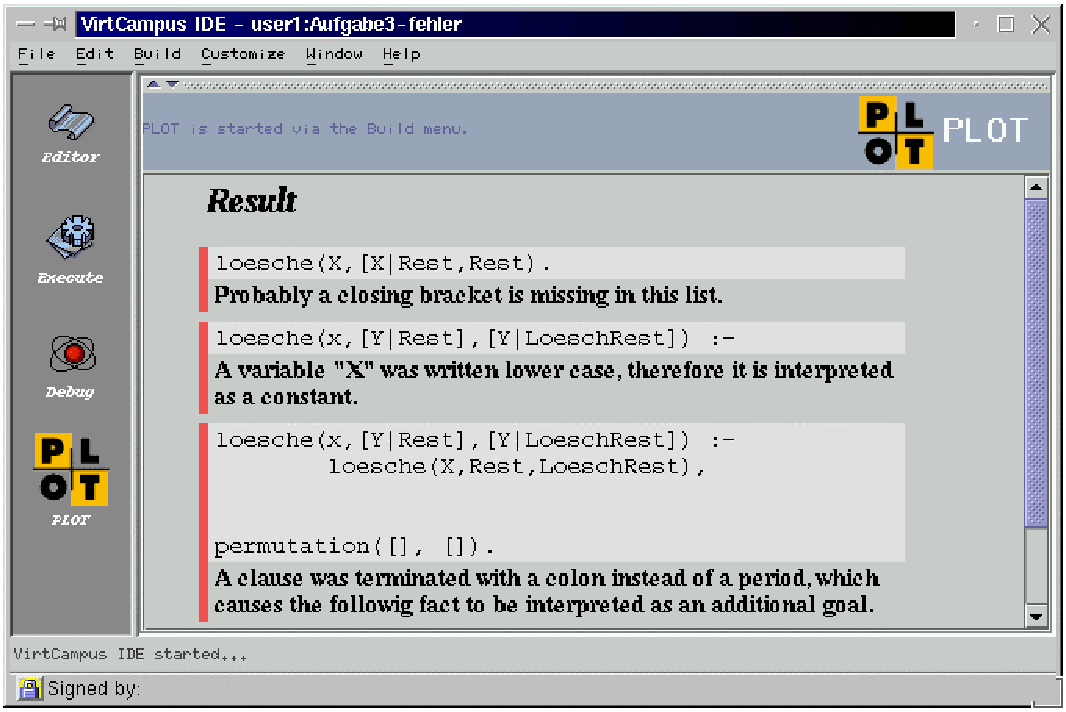

Figure: Screen-shot: Result of the intelligent analysis (PLOT)

The VC PROLOG tutor is to support both students and tutors with respect to their individual tasks. A student may choose the degree of the system's support. This ranges from practicing with predefined exercises getting full system support involving detailed syntactic and semantic error explanation to a more self directed use of the programing environment. While the semantic error explanation facilities are limited to a certain complexity of PROLOG, other offerings like access to the prolog ontology (conf. 3.2) may be used at any level. Thus, advanced learners may use the system as programing environment with convenient interfaces for accessing lecture notes and manual.

On the tutor's side the system offers support with respect to course and user management. Besides, it offers authoring tools for assignments and example solutions and supports the correction of assignments.

The present system does not only provide the means for learning PROLOG but may be regarded as a general framework for any kind of Internet educational environment with integrated course management. We accomplished a prototype of a LISP tutor to demonstrate that the VC tutoring system may be adopted to different domains.

3.2 System's architecture

The goals of the VC project influence on the architectural design decisions:

- The tutoring system is to address a large number of users who are to learn PROLOG. The prerequisites for using the system are to be kept to a minimum, i.e. no explicit computer science background is necessary. A user shall be enabled to concentrate fully on the PROLOG- learning task. Thus, user actions related to software installing and system configuration have to be avoided.
- It should be possible for students to use the system from different places (home, university network), thereby having access to all their data.
- A user shall be enabled to work with his familiar working space and applications.
- To make use of common PROLOG features such as defining and re-using predicates the system has to maintain one process for each user.
- Privacy of user data is an important issue. Therefor, the transfer of user data between machines has to be organized in a reliable and secure way.
- System administration and maintenance has to be quick and efficient. Irritations and problems due to different system versions are to be avoided.

These specifications require a *client-server* architecture [Ste90]. Communication between client and server can be achieved by different approaches, such as a HTML forms based client and a Common Gateway Interface (CGI) on the server side, a JAVA client or HTML on the clients side and JAVA SERVLETS on the server side.

The CGI based approach does not offer persistent communication channels between a server and a client. Matching between a client and a corresponding persistent application process would have to be managed by cookies, which can not be regarded as an ideal solution with respect to security. Furthermore, the user interface would be limited to HTML features, whereas JAVA offers more facilities to create an interactive environment.

Thus, we designed the VC PROLOG Tutor with a JAVA client as user front end. This enables the user to access the system by familiar applications. The server side contains a server, written in ANSI-C, a POSTGRES database and the application process (such as PROLOG or LISP). The close communication between client and server results in a joint system which allows a user to access remote resources in the same way as a part of his own file system.

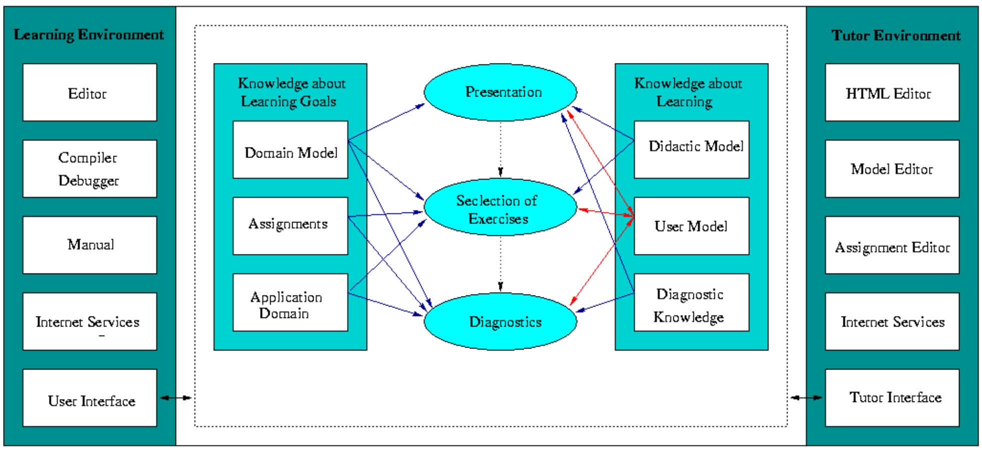

Figure: The VC Prolog Tutor. Data is represented as boxes, processes that operate on data are symbolized as ovals. Arrows pointing form data to processes symbolize that the processes are interpreting data. Arrows pointing from processes to data symbolize that these processes manipulate data. The interfaces for students and tutors are displayed on the right and left side of the outer frame.

4 Deployed technology for intelligent user support and user modeling

Web-based educational systems inherit their basic technologies from two kind of earlier systems: ITS and adaptive hyper-media systems [Bru99]. The ITS-related facilities include curriculum sequencing, intelligent analysis of student's solutions and interactive problem solving support. Adaptive navigation support and adaptive presentation are related to the adaptive hyper-media system part (conf. [KK97], [BC99]). In this section we will focus on technologies that are deployed in the VC PROLOG tutor to support students' interactions with the system.

4.1 Intelligent analysis of assignments

Basically, there are two different strategies of analyzing problem solving tasks, such as programing assignments: firstly, analysis of solutions (or partially accomplished solutions), secondly, analysis of the problem solving process. Both strategies are deployed by various intelligent tutoring systems to teach programming or certain abilities from this field [dBS87]. The first strategy has been employed in systems, such as LAURA [AL80], MENO-II [SOL83] and [Hon97], whereas BRIDGE [BS85], PROUST [Joh90] and Anderson's LISP tutoring system [ABCL90] are based on the second approach. Apart from these main approaches there are other strategies that may be considered to be a combination of both, such as [Web94], [Web96], [SW97].

The intelligent analysis of student's solutions in the VC PROLOG tutor was designed and implemented in the course of a one year students project [BGH+99]. Basically, it deploys the first strategy, since it is based on a comparison of the student's solution with one or more example solutions. Error analysis is accomplished in several steps by different modules. Firstly, a parser analyses the student's code. Syntax errors are marked and corrected. In case of several possible interpretations the analyzer will produce all alternatives.

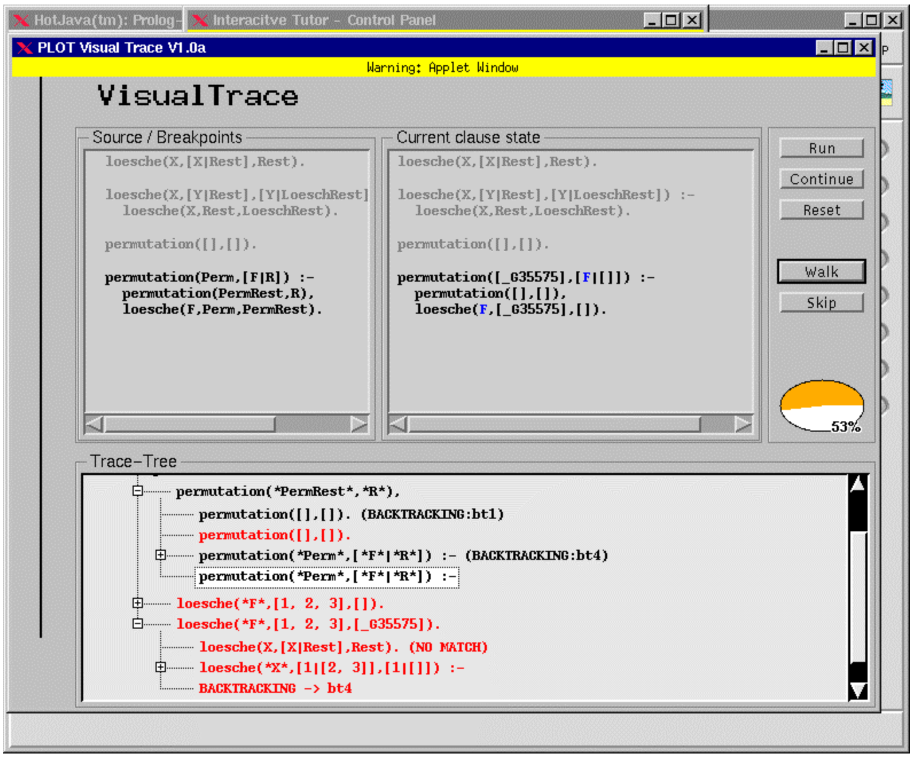

Figure: Screenshot: Graphic debugger

Secondly, the syntactically correct code is semantically analyzed, i.e. the meaning of the program is compared with the meaning of the example solution. Thus, it is to ensure that for each defined predicate there is a corresponding definition, that all loops terminate and that the students's program produces the same set of solution as the specimen solution.[1]

Thirdly, layout and structure of the student's code are compared against the example solution. Those modules are based on the assumption that the learner will obey to basic layout conventions. This may be helpful in those cases when several interpretations are possible.
Finally, the analyzer interprets the errors that have been located in the different modules and presents them to the user along with corrections and explanations. During the analysis the system generates information about the program that enables the graphic display of the running program in an environment similar to a debugger (see 3.3). Since it is a *post mortem analysis* with additional information the current binding of the variables can be displayed in each step. This helps the learner to visualize and understand (cf. [Bru94]) the way a PROLOG program works.

As mentioned above (conf. 2) the VC PROLOG tutor may offer help only to a limited extend, i.e. while PROLOG is used in a logical, non-procedural way. Thus, students have the possibility to request help from a human tutor via the system. This is similar to the *humans-*

[1] There are no iterative constructs like `for` or `while` loops in PROLOG, of course. Nevertheless, there is a class of recursive programs that correspond quite closely to iterative programs in conventional programing languages. Thus, we refer to recursion if we mention loops in PROLOG. Infinite loops, i.e. recursions that do not terminate, lead to program abortion due to space requirements, since a stack frame has to be maintained for every recursive call.

in-the-loop approach of [KMG99] since the system does include the option of human assistance.

4.2 Representation of educational material

As mentioned above, technologies for representing and presenting domain knowledge, become very important to guide the student through the available education material. Presently, adaptive hyper-books are a promising approach to present educational material (conf. [FNW98], [MCP+99]).
Since adaptivity depends on the "grain size" of the underlying domain model, we presently put the emphasis in our research in modeling the PROLOG domain in an ontology (conf. [IHL+99]). We have discussed our approach in an earlier paper [PTRG00], thus we will only give a short survey here. The entities in our ontology are related to concepts and skills which are fundamental for understanding and learning to write programs in PROLOG. For example, concepts like *terms, facts* or *clauses* are related to the syntax of PROLOG. *Recursion* and *unification* are basic for understanding the semantic of PROLOG. Whereas making use of *goal order* is an application of tactical knowledge and abilities like using an editor or compiling source code is part of pragmatic knowledge. Thus, syntax and semantic of a programing language, strategic and tactical knowledge, finally knowledge and skills related to the use of the programing environment have to be taken into account in programing teaching context [dBS87].

We have organized the domain knowledge in a concept lattice [GR96]. This offers a well-founded formalism to represent the dependencies between concepts. For facilitating intelligent problem solving support, concepts are additional related to skills (cf. [FDK+90]). Skills may be either defined in terms of knowing the intension and extension of a concept or in applying a concept to a task. [DG95] have shown how this approach may be employed for user modeling. We think that this approach will enable us to facilitate adaptive presentation of learning material with respect to the knowledge of the user.

4.3 A criterion for understanding

In any educational environment a criterion for *understanding* has to be defined. In this context are two questions relevant: 1.) does a learner *know* the concepts and techniques of an educational goal, such as PROLOG, and 2.) does he know how to *apply* his knowledge to a given task. We have chosen a pragmatic criterion for these considerations: if a student solves a specific problem with the strategies and techniques that are applied usually on this class of problems, i.e. acts within the PROLOG programing paradigm, he has reached this goal. Thus, we use a knowledge-level description [New82] to categorize the behavior of a student.
We use two information sources to judge students' understanding of PROLOG: url-tracking[2] to detect the visited PROLOG concepts from the lecture notes during the work on an assignment, secondly, the results from the intelligent analysis of assignments (see 3.1.). Url-tracking enriched with information from the JAVA-client offers an important insight how the competence state of a student is related to his interactions with the tutoring system. A student who performs good with respect to the assignments without browsing the educational material of the system has either already some knowledge about the domain or uses educational material from different sources. A learner who browses the lecture notes but does not work on assignments does not enable the system to trace his understanding process either. Students who use the educational material during their work on assignments enable the system to trace their assumptions on what parts of the learning material are relevant with

[2] Url-tracking is performed server-sided by interpreting the log files of the web server. It is based on cookies.

respect to the given problem. Thus, together with the analysis of the intelligent analysis of assignments it is possible to see how they achieved the transfer of the theoretical knowledge taken from the learning material. Thus, the system abilities to estimate the degree of understanding of a student are limited to those cases where data from both sources, url-tracking and accomplished assignments, is available.

5 Experiences with the VC PROLOG tutor

The VC PROLOG tutor has been used in programing courses in the summer term 1999 and in the winter term 1999/2000 during AI seminars. The course was attended by 80 students of computational linguistics and cognitive science. The VC PROLOG tutor was used during the seminars at the university and students were expected to use the system to solve their assignments.

During the terms we noticed a shift in the use of the system. The students who performed best installed PROLOG on their machines at home and preferred to work off-line, students who needed more help and coaching tended to use the system during the whole term.

Although the feedback of students and tutors was positive, we learned that some problems which play a minor role in the academic context may be nevertheless a serious hinderence in commercial settings. We expected by using APPLETS in web-browsers as user front-end to achieve independence of operating systems but a large class of problems is related to web-browsers and window managers *on* different operating systems. New versions of browsers often have an annoying lack of stability and any application based on web-browsers are highly dependent on them - and on the bandwidth of the underlying net. We solved this problem by installing JAVA on the local machines and using the APPLET-viewer. This was a drawback since we tried in our approach not to force a user to install additional software.

These experiences may lead us to the conclusion, that an architecture with JAVA client in a common web-browser may be risky for commercial systems. Maybe server-sided JAVA could be an alternative but we have no experiences with these modules so far.

5.1 Further work

So far, the information contained in the user model is mainly used for class monitoring. During the next months the modeling of concepts and skills for the ontology will be finished. Then we will put the emphasis of our work in deploying the user model for adaptive presentation and concept sequencing.

6 Bibliography

ABCL90

J.R. Anderson, C.F. Boyle, A.T. Corbett, and M.W. Lewis.
Cognitive Modeling and Intelligent Tutoring.
In J. Clancey and E. Soloway, editors, *Artificial Intelligence and Learning Environments*, pages 7 - 51. MIT-Press, Cambridge, London, 1990.
Reprinted from Artificial Intelligence: an International Journal, volume 42, number 1, 1990.

AL80

A. Adam and J.P. Laurent.
Laura, A System to Debug Student Programs.
AI, 15:75-122, 1980.

BC99

P. De Bra and L. Calvi.
Aha! An open Adaptive Hypermedia Architecture.
The new review of Hypermedia and Multimedia, 4:115 - 139, 1999.

BGH+99

C. Bönnen, J. Gerstenhöfer, N. Hartmann, A. Holzwarth, K. Kober N. Lenschow, and B. Meißner.
Plot: ProLog Online Tutor.
Technical report, Institut für Semantische Informationsverarbeitung, 1999.

Blo84

B.S. Bloom.
The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring.
Educational Researcher, 13(16), 1984.

Bru94

P. Brusilovsky.
Explanatory Visualisation in an Educational Programming Environment: Connecting Examples with General Knowledge.
In B. Blumenthal, J. Gornostaev, and C. Unger, editors, *Proceedings*, pages 202-212, Berlin, Heidelberg, New York, 1994. Springer.

Bru99

P. Brusilovsky.
Adaptive and Intelligent Technologies for Web-based education.
Künstliche Intelligenz, 4:19 - 26, 1999.

BS85

J. Bonar and E. Solloway.
Programming Knowledge: A Major Source of Misconceptions in Novice Programmers.
Human-Computer Interaction, 1:133-161, 1985.

dBS87

B. du Boulay and C. Shothcott.
Computers Teaching Programming: An Introductory survey of the field.
In R.W. Lawler and M. Yazdani, editors, *Artificial Intelligence and Education*, volume 1, pages 345 - 372. Ablex, 1987.

DG95

I. Düntsch and G. Gediga.
Skills and knowledge structures.
British Journal fo Mathematical and Statistical Psychology, 48:9 - 27, 1995.

FDK+90

J.C. Falmagne, J.P. Doignon, M. Koppen, M. Villano, and L. Johannesen.
Introduction to Knowledge Spaces: How to Build, Test and search them.
Psychological Review, 97(2), 1990.

FNW98
P. Fröhlich, W. Nejdl, and M. Wolpers.
Kbs-hyperbook - An Open Hyperbook System for Education, 10th World Conference on Educational Multimedia and Hypermedia (EDMEDIA 98), 20.-25. June 98, Freiburg, Germany, 1998.

GR96
B. Ganter and R.Wille.
Formale Begriffsanalyse.
Springer, Berlin, Heidelberg, New York, 1996.

Hon97
J. Hong.
An Intelligent Tutoring and Assessing System for Prolog Programming.
In B. du Boulay and R. Mizoguchi, editors, *Artificial Intelligence in Education*, pages 586-588, Amsterdam, Berlin, Oxford etc., 1997. IOS Press.

IHL+99
M. Ikeda, Y. Hayashi, J. Lai, W. Chen, J. Bourdeau, K. Seta, and R. Mizoguchi.
An Ontology - more than a shared vocabulary.
In *AI-ED 99 Workshop on Ontologies for Educational Systems*. 1999.

Joh90
W.L. Johnson.
Understanding and Debugging Novice Programs.
In J. Clancey and E. Soloway, editors, *Artificial Intelligence and Learning Environments*, pages 51 - 99. MIT-Press, Cambridge, London, 1990.
Reprinted from Artificial Intelligence: an International Journal, volume 42, number 1, 1990.

KK97
J. Kay and B. Kummerfeld.
User models for customized hypertext.
In C. Nicolas and J. Mayfield, editors, *Intelligent Hypertext: Advanced Techniques for the World Wide Web*, LNCS. Springer, Berlin, Heidelberg, New York, 1997.

KMG99
V. Kumar, G. McCalla, and J. Greer.
Helping the Peer Helper.
In S.P. Lajoie and M. Vivet, editors, *Artificial Intelligence in Education. Open Learning Environments: New Computational Technologies to Support Learning, Exploration, and Collaboration*, pages 325 - 332, Amsterdam, 1999. IOS Press.

MCP+99
T. Murray, C. Condit, J. Piemonte, T. Shen, and S. Kahn.
Meta-links - a framework and authoring tool for adaptive hypermedia.
In S.P. Lajoie and M. Vivet, editors, *Artificial Intelligence in Education. Open Learning Environments: New Computational Technologies to Support Learning, Exploration, and Collaboration*, pages 744-746, Amsterdam, 1999. IOS Press.

New82
A. Newell.
The Knowledge Level.
Artificial Intelligence, 18:87-127, 1982.

PTRG00
C. Peylo, W. Teiken, C. Rollinger, and H. Gust.
An Ontology as domain model in a web-based educational system for prolog.
In *Proceedings of the 13th Flairs Conference*. AAAI Press, 2000.
to appear.
SOL83
E.M. Soloway, B.P. Woolf, E. Rubin, J. Bonar and W.L. Johnson.
MENO-II: An AI-based programming tutor.
Journal of Computer-based Instruction, 10 (1): 20-34, 1983
Ste90
R. Stevens.
Unix Network Programming.
Prentice Hall, Englewood Cliffs, New Jersey, 1990.
SW97
M. Specht and G. Weber.
Kognitive Lernermodellierung.
Kognitionswissenschaft, 6(4):165-176, 1997.
Web94
G. Weber.
Fallbasiertes Lernen und Analogien.
Psychologie Verlags Union, Weinheim, 1994
Web96
G. Weber.
Episodic Learner Modeling.
Cognitive Science, 20: 195-236, 1996.
Wen87
E. Wenger.
Artifcial Intelligence and Tutoring Systems.
Morgan Kaufmann, Los Altos, California, 1987.